Impact of Layer-1 Characteristics on Scalability of Layer-2 Semi-Hierarchical Payment Channel Networks*

Marco Benedetti, Francesco De Sclavis, Giuseppe Galano, Sara Giammusso, Antonio Muci and Matteo Nardelli

Bank of Italy

Abstract

Assessing the scalability of a payment channel network is crucial as it directly impacts the efficiency and user experience of blockchain-based payment systems used in high volumes transaction environments, such as Central Bank Digital Currencies (CBDCs). In this work, we present an analysis of the interaction between a Layer2 Semi-Hierarchical Payment Channel Networks (SH-PCNs) and its Layer1 blockchain technology. In particular, we focus on how blockchain features, such as latency, throughput and congestion, affect the payment success rate within a SH-PCN, in a scenario in which routing nodes try to keep the state of their channels balanced by exchanging off-ledger and on-ledger liquidity, a technique known as submarine swaps. We use a Parallel Discrete Event Simulator (PDES) for SH-PCNs, in which we incorporate a blockchain component to simulate the dynamics of a limited block size and non negligible block time. We simulate various scenarios, defined by: the total amount of liquidity locked in the channels, constant and variable Layer2 payment load, Layer1 blockchain congestion conditions and channel rebalancing strategies adopted by nodes. The experiments reveal how blockchain parameters like congestion rate significantly influence PCN performance metrics such as payment success rate. We conclude that, when designing a PCN for scalability, it's important to take into account that Layer1 characteristics affect the optimal allocation of liquidity that is needed to match the target performances.

1. Introduction

The groundbreaking introduction of blockchains to support the exchange of crypto-assets has also generated a huge hype regarding its adoption for supporting world-wide efficient retail payments and, as such, for the design of Central Bank Digital Currencies (CBDCs). Today, retail instant payment systems manage a very high load of transactions, in the order of 10^4–10^5 TPS (e.g., [1, Sect. IV-A]), to be settled within 10 seconds [2]. Besides performance, CBDCs may also include further business and technical requirements, such as privacy guarantees, small or no fees for citizens, a cap on the amount of liquidity users can amass, and the possibility to be usable by unbanked people.

DLT'24: 6th Distributed Ledger Technology Workshop, May 14–15, 2024, Torino, Italy

*All views are those of the authors and do not necessarily reflect the position of Bank of Italy.

marco.benedetti@bancaditalia.it (M. Benedetti); francesco.desclavis@bancaditalia.it (F. De Sclavis); giuseppe.galano2@bancaditalia.it (G. Galano); sara.giammusso@bancaditalia.it (S. Giammusso); antonio.muci@bancaditalia.it (A. Muci); matteo.nardelli@bancaditalia.it (M. Nardelli)

0009-0004-1318-3878 (F. De Sclavis); 0009-0008-3251-6606 (G. Galano); 0009-0009-9355-3416 (S. Giammusso); 0000-0002-9519-9387 (M. Nardelli)

In our previous works [1, 3, 4], we started to investigate whether a blockchain can be an appropriate technological choice to implement a (hyphotetical) CBDC. As expected, ingesting and settling a realistic transaction load in a timely manner is challenging, especially in a fully decentralized system that uses a blockchain to store its global state. Therefore, in [1], we embraced the off-ledger paradigm, whereby scalability is achieved by an additional *payment channel network* (PCN), a second layer built on top of the actual blockchain [5]. It results a two-layered system where the first (wholesale) layer, i.e., the blockchain, exhibits high integrity, availability, fault-tolerance, and verifiability of monetary exchanges; and the second (retail) layer, i.e., the PCN, accommodates fast-payments, cash-like levels of privacy, and handles wallet caps. To sustain the payment load and map technical PCN roles with the business roles of the 3-tier banking system of the current monetary system [6], we explored a family of PCNs with a partially constrained topology, called "*Semi-Hierarchical Payment Channel Networks*" (SH-PCNs)—see Sect. 2.2. Besides opening and closing channels, PCNs send transactions to the underlying blockchain every time the channel should be reconfigured, e.g., to balance its capacity (see Sect. 2.3). Therefore, some operations in the PCN need to wait for the transaction confirmation on the blockchain before completing. In [1], we conducted our analysis on PCNs with a strong—yet commonly adopted (e.g., [7, 8, 9, 10])—simplifying assumption on the interaction between the first and second layer: i.e., we assumed that each on-ledger request from the PCN would be satisfied with a constant delay by the blockchain. We all know that in real life, things turn out very different.

Several approaches assume that on-chain transactions can be satisfied (i.e., included in a block) after a constant, predictable delay equal to the block time of the blockchain (e.g., [9, 7, 8, 10]). To the best of our knowledge, so far, there is no tool for investigating how different working conditions of the blockchain influence performance of second layer systems, such as a PCN.

In this paper, we investigate the interaction between blockchain and PCNs, by specifically focusing on its impact on the payment success rate, i.e., the probability that a payment is successfully forwarded from the sender to the recipient in the network. To extensively evaluate such a subtle but pivotal interaction, we resort to simulations. In particular, we extend *CLoTH-over-ROSS*[1], our Parallel Discrete Event Simulator (PDES) [11], to introduce a model of the blockchain and of its interaction with the PCN. Basically, our PDES is a porting of CLoTH [12], an open-source sequential simulator of a Lightning Network (LN) implementation[2] widely used in recent works (e.g., [13, 14, 15]), on ROSS [16]. The resulting simulator represents a valuable tool to gain interesting insightful regarding the impact of layer 1 on layer 2, where we can readily see the effect of blockchain working conditions (e.g., congestion) on performance achievable off-ledger by the PCN. The key contributions of this paper are as follows.

- We present a model of the blockchain suited to explicitly represent its interaction with a layer-2 PCN (Sect. 3). Driven by the use case needs, we consider the blockchain as a single entity of the system that stores incoming transactions and periodically feeds them in a new block. Different representations, serving different purposes, can be easily taken into account;

[1]The source-code of CLoTH-over-ROSS will be available when we submit the camera-ready version of this paper: https://bancaditalia.github.io/itcoin/

[2]https://github.com/lightningnetwork/lnd

- We run an extensive set of experiments, aiming to investigate how the blockchain impacts performance of the PCN in terms of channel rebalancing, locked liquidity in channels, and payment success rate, when different blockchain parameters change, e.g., its congestion rate (Sect. 4).

2. Background

2.1. Permissioned Blockchain for Central Bank Digital Currency

In [4] we consider a Bitcoin-like permissioned blockchain for a CBDC, where a set of known *miners* validates transactions to be added to the blockchain. Each miner is operated by one member of a federation of independent, geographically distributed, trusted actors[3], that are called *validators*. Validators run a Byzantine fault tolerant consensus protocol and collectively publish a new valid block every *target block time*, which in our examples is set to one minute. As in most approaches that require geographically distributed Byzantine consensus among a dozen of nodes, also the blockchain we consider suffers from scalability issues. Therefore, in [1], we explore the possibility to exploit Layer-2 solutions to safely exchange instant payments off-ledgers, and in particular a specific topology of PCN [17], that we call Semi-Hierarchical PCNs, or SH-PCN for short.

2.2. Semi-Hierarchical Topologies in Payment Channel Networks

In a PCN, two nodes create a bilateral *payment channel* by locking some amount of liquidity, called *channel capacity*, into a 2-of-2 multisig UTXO, using a single *on-chain* funding transaction. The sum of nodes' balance in the channel can never exceed the payment channel capacity.

After the channel is created (i.e., the funding transaction is confirmed), the nodes can exchange multiple, instant, *off-chain* payments, which update the balances of nodes and eliminate the need to constantly execute on-chain transactions. The safety of payments within the channel is ensured through the pre-funding and a cryptographic mechanism, called *channel revocation*. If a cheating attempt by one party is detected, the other is entitled to claim the entire channel capacity. A comprehensive description of PCNs can be found in [5, 17].

The most popular PCN implementation, i.e., the Lightning Network (LN), does not constraints how PCN node should open channels with one another. Therefore, different network topologies can be build or can emerge, ranging from star-like to scale-free (as appears to be the LN, e.g., [18, 19]). Each topology is characterized by its peculiarities in terms of efficiency, fault-tolerance, security, and privacy (e.g., [18, 20]). In [1], we investigate a special family of PCNs, named Semi-Hierarchical PCN (SH-PCN, for short), and analyze how such a network topology can sustain a real payment traffic.

To understand the rationale behind a SH-PCN, we briefly introduce the business roles of a monetary system that we imagine to be mapped on different technical roles in the PCN. The current monetary system—which is the target deployment environment for our solution—is based on a *3-tier banking system* [6], where: the CB is at the top; a set of authorized intermediaries (e.g., commercial banks) are in the middle; retail users (e.g., citizens and merchants) are at the

[3]We target settings where the number of trusted actors is expected to be between 4 and $\approx$ 20.

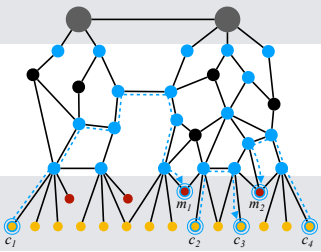
Tier	Technical role	Business entities	Size (magnitude) actual / simulated	Sample SH-PCN	Sub-network topology	Channel capacities	Rebalancing strategy
1	RSP, MH	Central Banks	10^1 / 10^0		Structured, hierarchical	"infinite" capacity	N/A
2	LSP CS	Commercial Banks / Financial Institutions	10^4 / 10^1 10^3 / n.a.		Free-form, organised by *market operators*	Allocated & optimised by *market operators*	Just-in-time rebalancing via *submarine swaps*
3	End-Users: payers, payees	Retail users: merchants citizens	10^6 / 10^3 10^8 / 10^5		Fixed topology (*Policy* decision)	Fixed capacity (*Policy* decision)	Just-in-time (*reverse*) *waterfall*

Figure 1: Main features of a SH-PCN, plus three sample payment routes: One *Point-of-Sale* payment from retail user c_4 (a citizen) to merchant m_2; one *Peer-to-Peer* transaction from c_2 to c_3; one *eCommerce* route from c_1 to a foreign merchant m_1.

bottom. Fig. 1 represents how these (business) roles are mapped onto different (technical) roles in the SH-PCN, i.e., monetary hubs, lightning service providers, custodian services, and end users. The hierarchical anatomy of the banking system is reflected in the (semi) hierarchical topology of the network. In a nutshell, these technical roles are the following.

- **Monetary Hubs (MHs)** operate nodes with considerable liquidity, which are largely/fully interconnected. They are managed by a Central Bank (CB) or a set of CBs.
- **Lightning Service Providers (LSPs)** freely open channels towards each other, forming a "small world"-like network. They are managed by authorized intermediaries, and offer service to EUs by opening channels with them.
- In addition to off-chain liquidity, EUs can own one or more accounts at **Custodian Services (CSs)**, e.g., banks or exchange. CSs are connected to the LSP network. The monetary interactions between EUs and CSs happen out-of-band and via LSPs.
- **End-Users (EUs)** access the network connecting to one or more LSPs. EUs only open channels with authorized LSP(s).

Our SH-PCN is a composition of various graph models, each representing the connections (i) between different categories of nodes, and (ii) within the same tier:

1. MHs are interconnected in a clique (each channel has capacity of €500 million), and guarantee network connectivity;
2. Links between MHs and LSPs are generated by dividing the LSPs into one subset per MH. Then, a channel is established from each MH to every LSP in its subset. The subset sizes are log-normally distributed ($\mu = 0$ and $\sigma = 1$); a range of channel capacities for this subnetwork is explored in the experiements.
3. Links between LSPs are modeled using a Watts-Strogatz graph [21] ($k = 4$ and $p = 0.1$); also in this subnetwork, experiments explore various channel capacities.
4. Channels between LSPs and EUs are modeled just like in (3), albeit with *fixed* capacities, which represent the cap on users' wallets (€3000). Channels linking LSPs with merchants vary based on the merchant's size: small (S), medium (M), and large (L) merchants are assigned capacities of €5k, €50k, and €500k, respectively.

2.3. Channel Rebalancing Techniques in the SH-PCN

Forwarding a payment updates the channel balance, as the payments requires one end to give funds to the other end of the channel. Therefore, the performance of SH-PCNs subject to a load degrades over time: Channels become more and more unbalanced, thus preventing them to participate in the routing of an increasing percentage of payments. Rebalancing techniques improve channel lifetimes while minimizing locked liquidity (e.g., [8, 9]). In [1], we describe three rebalancing techniques: an on-chain solution, i.e., submarine swap, and two off-chain solutions, i.e., waterfall and reverse waterfall.

Submarine swaps between LSP/MH nodes. A *submarine swap* is an exchange of a given amount of some on-chain asset with the same amount of the off-chain form of the same asset. Hash Time Locked Contract (HTLCs) guarantee the atomicity of the transaction. Since submarine swaps involve *on-chain* transfer, they take time (at least the confirmation time of the HTLC) and possibly require a fee. Importantly, the transactional capacity of the blockchain limits the number of submarine swaps per unit of time that can be perform in a PCN. This is the focus of this paper, where we investigate the impact of the layer-1 blockchain on layer-2 PCN.

Waterfall rebalance. The *waterfall* [22] mechanism allows end-users—in particular those having high inbound traffic (e.g., merchants)—to always be able to get paid, even if the amount P to be received raises the user balance B above the channel capacity C (wallet cap). This is achieved by automatically depositing the amount $D = \max(B + P - C, L_D)$, where L_D is the minimum amount the user is willing to deposit, to a linked CS account. When the LSP receives a payment to forward but the end user's channel does not have enough outbound liquidity, the LSP *notifies* the user about the incoming payment, and delays the payment forwarding until the expiration of a timeout. The user requests a real-time deposit to their CS, and sends the deposit via the same LSP, thus rebalancing the channel. If the channel is successfully rebalanced within the timeout, the LSP forwards the payment to the user; otherwise, the payment fails.

Reverse waterfall rebalance. This functionality allows retail users—in particular those having high outbound traffic (e.g., citizens)—to automatically fund a payment channel before making transactions too large for its current state. If there are insufficient funds in the channel to cover the amount P, users could request a withdrawal, thus taking out some money from their CS account. The withdrawal amount is $W = \max(L_W - B, P - B)$, where L_W represents a minimum amount the user is willing to keep in its wallet for future use. Once the withdrawal has transferred liquidity from the linked CS to the channel, the user can send the payment.

3. Simulating Blockchain and its PCN

3.1. PCN Simulator Architecture

In [1, 11], we described a parallel PCN simulator architecture that we used to analyze the performance of PCNs in the context of CBDCs. The simulator extends CloTH [12], an open source sequential discrete event simulator for PCNs that implements source-based path finding and the mechanism of HTLC. We rebuilt CLoTH on ROSS [16], a general purpose framework for parallel discrete event simulations that facilitates the simulation of large-scale networks using multiprocessing and a distributed memory architecture. We presented simulation results

regarding payment success rate, payment completion time, and liquidity costs in different generated SH-PCN topologies (see Fig. 1), which mimic a hypothetical CBDC implemented within a three-tier banking system. In our simulations, the payment load is independently generated by each EU, and is designed in a way that globally it achieves an emerging network behavior that follows real-world payment statistics [23]. A detailed description of the PCN-related part of the simulator can be found in [1, 24].

3.2. Extending the PCN Simulator with a Blockchain Component

A missing piece in the previously described parallel simulator architecture is the detailed analysis of the interaction between the PCN (as a second-layer solution) and its underlying layer-1 blockchain. Indeed, in several scenarios users interact with the underlying blockchain through on-chain transactions, in order to ensure liquidity or security of payment channels. These include: (i) opening of channels, in which the channel capacity gets locked; (ii) closing of channels, that distributes the locked funds to the parties according to the final state of their off-chain transactions; (iii) channel splicing, an advanced feature that allows users to adjust the amounts of funds locked in a payment channel without closing it; (iv) dispute resolutions, in case there is a disagreement between parties about the state of the channel; (v) settlement of HTLCs, that are primarily off-chain interactions but may lead to on-chain transactions in cases when one party is unreachable and does not cooperate to the HTLC consolidation; and (vi) submarine swaps, that enable users to transfer funds between on-chain and off-chain wallets without closing existing channels.

Specifically, the impact that *block time* and *block size* limitations can have on the simulation results is a crucial aspect that warrants further exploration. These two parameters have a direct impact on transaction latency and on the overall blockchain throughput. The former (transaction latency) refers to the time taken for a transaction to be confirmed and added to the blockchain and can significantly influence the payment success rate and liquidity efficiency of PCNs. The latter (blockchain throughput) refers to the number of transactions processed per unit of time and may become a bottleneck especially when multiple channel actions need to be performed simultaneously. This limitation can lead to increased transaction fees and delayed channel operations, thereby affecting the PCN's ability to handle high volumes of transactions efficiently.

For these reasons, we introduce a lightweight model of the blockchain, in which we abstract away all details related to its network, consensus, and application protocol. In particular, we focus on the ability of the blockchain to receive transactions, collect them in a transaction pool (mempool), and periodically publish a new block. We model the blockchain as a single process that manages the chain data structures and mempool. The chain represents the sequence of all confirmed blocks, whereas the mempool includes transactions waiting to be included in a block. The blockchain process creates a new block every `block time`, that is a simulation parameter we set equal to 1 minute in Sect. 4. The blockchain throughput is controlled by two parameters, called `block size` and `congestion rate`, representing respectively the number of transactions that can be included in a block and the percentage of the block size that cannot be used during the simulation. We introduce the `congestion rate` in order to account for on-chain transactions that are not in the scope of the simulation. We consider that the simulation

can use 1 - `congestion rate` of the blockchain space, as we consider the remaining size of the block to be filled by other blockchain transactions (e.g., wholesale on-chain payments).

Algorithm 1 Blockchain Process

```
Require: MEMPOOL → []
Require: BC → []                                          ▷ Blockchain
  upon event BC_TX_BROADCAST do                           ▷ New transaction tx received
    tx ← received transaction
    if tx is valid then
      Add tx to MEMPOOL
    end if
  upon event BC_TICK do                                   ▷ Time to forge a new block
    B → []                                                ▷ New block
    while B is not full and MEMPOOL is not empty do
      Get tx from MEMPOOL
      Add tx to B
    end while
    Add B to BC
    Notify sender and receiver of all txs in B
    Trigger new BC_TICK event according to exp(block time)
```

Algorithm 1 reports the pseudo-code of the blockchain process. The blockchain process reacts in response to two simulation events: `BC_TX_BROADCAST` notifies that a new transaction has been submitted to the blockchain, `BC_TICK` event sets the blockchain timing and ensures that a new block is added to the chain when expected. A new `BC_TICK` event is triggered according to an exponential distribution having `block time` as mean. In the simulation of the blockchain component, we made two simplifying assumptions. First, on-chain transactions and blocks are received by the blockchain miner and PCN nodes respectively after a (small) network delay, and no delays are introduced by gossiping algorithms for information dissemination. We extract network latencies from a gamma distribution $\Gamma(6.4, 4.35)$ estimated on the average daily AWS network latencies[4] among European regions. Second, the mempool is a *first-in-first-out* (FIFO) queue, and the sooner the blockchain receives a transaction, the sooner it will be included in a block. We do not implement fee estimations when sending transactions and a transaction selection algorithm based on fees when selecting transactions from the mempool.

3.3. Submarine Swaps in a Constrained Layer1 Environment

As described in Sect. 2.3, we implement three channel rebalancing channels: i.e., submarine swaps, waterfall rebalance, and reverse waterfall rebalance. Only the first one requires on-chain transactions, since the others can be performed entirely in the off-chain layer (i.e., in the PCN). In this section, we briefly describe the simulation of submarine swaps in an environment with a blockchain having limited throughput and high latency.

[4]Source: https://www.cloudping.co

A submarine swap is initiated by either an LSP or an RSP in response to a FORWARD PAYMENT event when a payment about to be forwarded would result in the channel balance exceeding a predefined *unbalancedness* threshold; When a node prepares to forward a payment, it checks the unbalancedness of the incoming channel. The incoming channel would result in having an increase in the local balance of the node if the payment is successfully forwarded. The unbalanceness of the incoming channel is defined as the ratio between the `local balance` and the `channel capacity`: an unbalancedness greater than 0.5 means that the channel liquidity is more concentrated on the side of the local node, and the upcoming payment is going to make things even more unbalanced. If the unbalancedness exceeds the *Submarine Swaps Threshold* (or simply Swap Threshold), the node attempts to initiate a submarine swap for the incoming channel. The happy path process is the following: (i) the node sends a SWAP_REQUEST to the channel counterparty; (ii) the counterparty initiates the swap sending a PREPARE_HTLC transaction to the blockchain; (iii) once the prepare transaction is confirmed, the node sends a submarine off-ledger payment to the counterparty, via the channel that parties aim at balancing; when the submarine payment is completed, the unbalancedness of the channel is reduced; and (iv) upon receipt of the submarine payment, the node learns the secret preimage used by the counterparty to lock the funds on the blockchain and can broadcast the CLAIM_HTLC transaction to the blockchain. Algorithm 2 summarizes how a submarine swap is simulated.

Algorithm 2 Submarine Swap simulation algorithm

Require: `Swap Threshold`>0.5
  upon event Forward Payment **do**
    if Incoming Channel Unbalancedness ≥ `Swap Threshold` **then**
      Forward Submarine Swap Request
    end if
  upon event Swap Request **do**
    Broadcast `Prepare HTLC`
  upon event Blockchain Transaction `tx` Received **do**
    if `tx` is of type `Prepare HTLC` **then**
      Send submarine payment
    else if `tx` is of type `Claim HTLC` **then**
      Delete swap
    end if
  upon event Receive Payment Success **do**
    Broadcast `Claim HTLC`

In the simulation of the submarine swap component we made two simplifying assumptions. First, a node receiving a swap request always accepts the swap and forwards the prepare HTLC transaction. Second, we do not handle submarine swap timeouts.

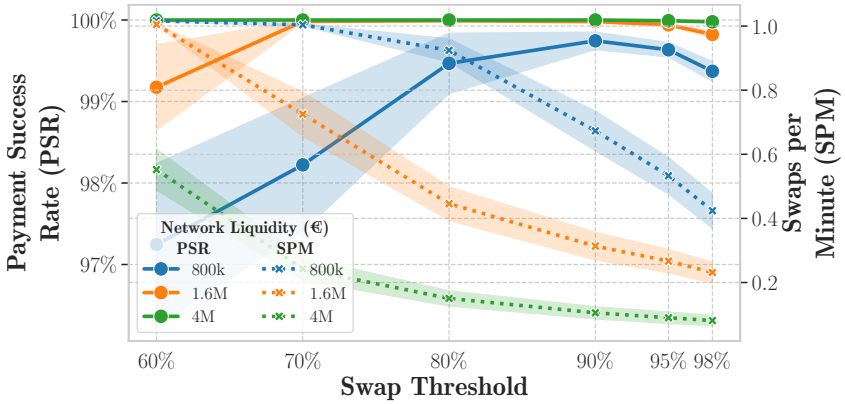


Figure 2: Effect of different submarine swap thresholds on the payment success rate and the overall number of swaps per minute performed by the PCN.

4. Evaluation

We generate and simulate 10 random SH-PCNs whose size is proportional to 4 countries—France, Germany, Italy, and Spain—with 4 MHs, 40 LSPs, and 44k EUs. For each SH-PCN, we simulate a stream of payments for 24 hours (of simulated time). In the following, we present and analyze the *mean* and *standard deviation* of the simulation results.

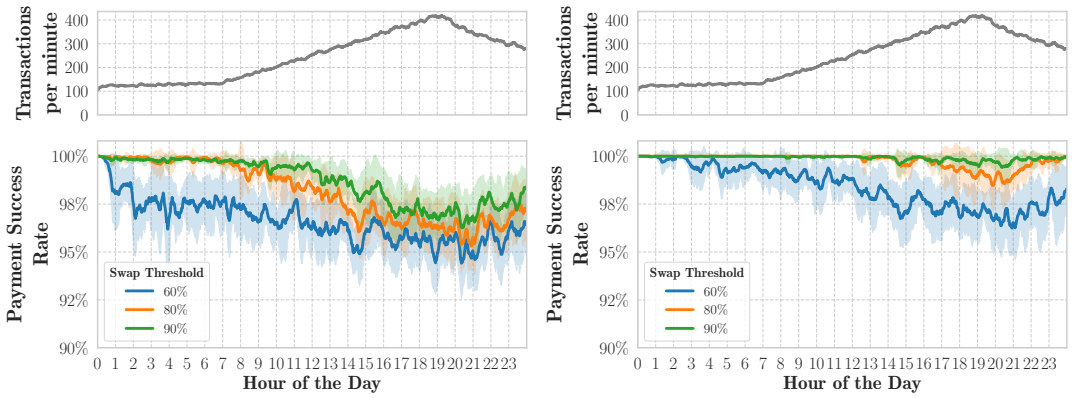
We present three main sets of experiments. First, we analyze the impact of the rebalancing policy on the payment success rate in a setting of constant payment generation rate. Second, we investigate what happens when the payment generation rate changes over time. Finally, we investigate the impact of blockchain congestion on the performance of the PCN.

4.1. Effect of Rebalancing Strategies on Payments

As detailed in Sect. 3, performing submarine swaps involves on-chain transactions. Different policies can be devised to trigger such an operation. Note that performing submarine swaps too soon leads to a higher volume of locked liquidity; conversely, performing them too late would results in a reduced payment success rate—as it increases the probability of having an unbalanced channel that is not ready to forward a payment.

In this section, we evaluate the effect of the threshold-based policy to trigger swaps presented in Sect. 3.3. We consider a fixed blockchain `congestion rate` equal to 50%, i.e., PCN-related transactions can occupy up to 50% of the `block size`. The `congestion rate` accounts for on-chain transactions that are not in the scope of the simulation. Fig. 2 reports the payment success rate and the number of submarine swaps per minutes as the policy threshold changes; we evaluate also the effect of different total network liquidity configurations, i.e., the liquidity locked in all MH-LSP and LSP-LSP channels of the network[5]. First of all, we observe a general trend whereby higher network liquidity results in higher payment success rate; this is a rather straightforward result as higher channel capacity ensures their ability to forward payments. Moreover, when a higher liquidity is locked, the PCN overall performs a sensibly lower number

[5]We configure the network so that MH-LSP channels lock twice the liquidity of LSP-LSP channels.



(a) Network Liquidity: 800k (b) Network Liquidity: 1.6M

Figure 3: Varying load of payments. Impact on payment success rate when different thresholds and network liquidity configurations are considered. The rolling average value of payment success rate in a window of 15 minutes over the entire 24 hours of simulated time is presented.

of submarine swaps. This depends also on the stream of payments we are considering: although they cover 24 hours, they do not move enough balance to trigger channel replenishing operations (i.e., submarine swaps). Three further observations can be made. First, submarine swaps enables the PCN to sustain continuous payments while keeping a high payment success rate. Independently from the amount of locked liquidity, Fig. 2 shows that at least a swap every 15-20 minutes must be performed to obtain 100% of success rate even with 4M of network liquidity. Second, the threshold-based policy help prevent the no-balance available errors in payment forwarding. Specifically, we can see that when the swap threshold increases up to 90%, the payment success rate increases (with network liquidity of 800k) or stays fixed to 100% (with liquidity greater or equal to 1.6M). Interestingly, higher values of the threshold result in a lower payment success rate: this is due to dynamics of the system where performing a swaps takes much more time than the payment time-out (set to 10 s to let us consider the payment as "instant" [2]). Third, low values of the threshold reduce the payment success rate, e.g., see the success rate with swap threshold equal to 60% and network liquidity equal to 1.6M. This sounds counter-intuitive, although we can observe that lower values of the threshold increase the number of on-chain transactions that more quickly fill the blockchain mempool. Such a high volume of requests delays swap requests, which cannot terminate in time to prevent no-funds error in payment forwarding.

Fig. 3 summarize results from a second set of experiments, where we change the payment load over time (as depicted in the upper part of the figure). By comparing Fig. 3a and Fig. 3b, we readily see that increased network liquidity requires fewer swaps (as seen before). When liquidity is 800k, even with a 90% of swap threshold, the payment success rate decreases as soon as the overall rate of payments starts to increase. Lower thresholds worsen performance. The situation is sensibly better with 1.6M of network liquidity. When the threshold is 90% or 80% the network can withstand up to 300 transactions per minute with basically no impact on the payment success rate. Note that when the threshold is 60%, the variability of performance

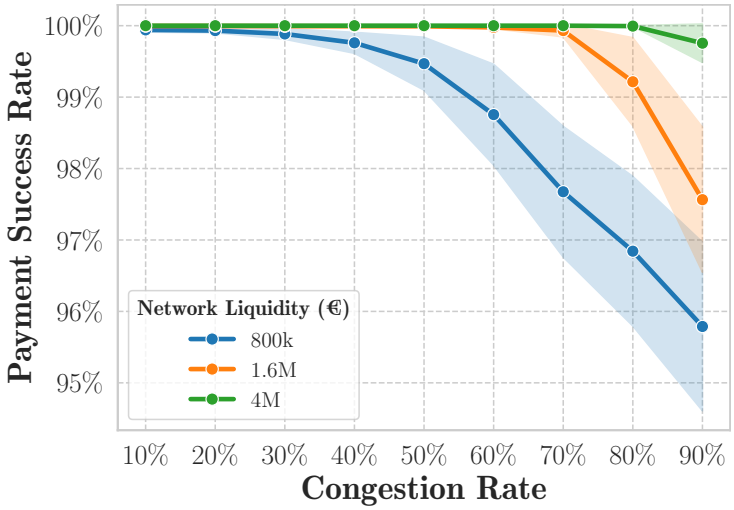


Figure 4: Effect of different blockchain congestion rate on the success of payment in the PCN.

is high indicating that the system is not able of correctly managing the load; observe how the success rate decreases even when the payment rate is constant in the left-hand side of Fig. 3b.

4.2. Effect of Blockchain Congestion on PCN Performances

In the previous section, we showed how the PCN performance depends on the blockchain: rebalancing a channel on-chain imposes long waiting times that can also result in payment failures. In this set of experiments, we investigate the impact of different blockchain congestion ratio on PCN performance. In particular, we fix the swap threshold to 80% and analyze the effect of different congestion rate of the blockchain on payment success rate. Fig. 4 summarizes results. We observe that, as expected, higher blockchain congestion prevents the PCN to timely perform submarine swaps, which can be included in a block after some `block time`. As also previously discussed, higher network liquidity requires fewer swaps; hence, the payment success rate stays high even with high blockchain congestion. Nonetheless, higher network liquidity implies higher costs that LSPs want to prevent. With 1.6M of network liquidity, the PCN can tolerate up to a blockchain congestion of 60-70% before a noticeable decrement in the payment success rate.

An in-depth evaluation of the threshold-base exchange threshold is worth investigating, it is beyond the scope of this work.

5. Related Work

5.1. Lifespan of Payment Channels

Although PCNs represent an interesting approach to improve scalability of blockchains, they are not free of drawbacks. As also shown in Sect. 4, channels get unbalanced while forwarding payments. Shabgahi et al. [9] propose a model to predict the expected time for a channel to get

unbalanced, considering its centrality and its initial balance. Basically, two main approaches emerged to replenish channels and extend their lifespan, namely via on-chain transactions or via in-place rebalancing (e.g., by introducing ad-hoc routing strategies). Approaches that resort to on-chain transactions (e.g., [7, 8, 9]) usually do not explicitly model the limits imposed by the blockchain and, often implicitly, assume that the blockchain can always satisfy rebalancing requests in a constant time interval (as we also did in [1]). To the best of our knowledge, Papadis and Tassiulas [10] propose the most detailed model of rebalancing through on-chain transactions. This model is then used to define a reinforcement learning based policy that proactively performs submarine swaps aiming to maximize profit from blockchain and PCN fees. The authors elegantly capture the different timescale at which rebalancing decisions and user transactions are performed. Nonetheless, for sake of tractability, they also assume that the blockchain introduces a constant delay to confirm all on-chain transactions.

We acknowledge that studying the constraints imposed by a blockchain subject to varying loads is complicated; it cannot be easily captured analytically. Therefore, we resort to simulations.

5.2. Simulating Blockchains and PCNs

Although deeply intertwined, to date there are no simulators that focus on the blockchain-PCN interaction. We first briefly look at blockchain simulators, and then summarize the key tools for simulating PCNs and, in particular, the LN.

5.2.1. Simulators of Blockchains

The popularity of blockchains as well as its complexity lead to the development of a rather large number of simulators (e.g., [25, 26, 27]), with nearly half of them available in open-source (e.g., [28, 29, 30, 31, 32]). They model a wide range of blockchain features ranging from network, consensus, data, execution and application layers. Paulavičius et al. [33] and Albshri et al. [34] present an extensive literature review on blockchain simulators, and provide an interesting comparison among them along different dimensions, including the supported features. Differently from our approach that relies on an established PDES, all of these simulators are sequential discrete event simulators. They do not exploit advanced simulation features, such as load distribution, optimistic event scheduling, which might turn out to be very useful when simulating large networks. To the best of our knowledge, all of these simulators do not explicitly consider second layer systems, and they do not easily allow modeling and evaluating other systems that depends on blockchain to operate.

5.2.2. Simulators of PCNs

In this context, simulations have been widely used, e.g., to evaluate policies related to channel design, rebalancing, routing, and privacy of PCNs. Although several PCN simulators exist (e.g. [35, 36, 37]), a clear comparison among them is lacking. Research efforts typically build their own simulator to evaluate specific features, therefore most of them—unfortunately—appear not

to be actively maintained (e.g., [35, 38]). One of the LN developer open-sourced a simulator[6] that focuses on the LN gossiping protocol. Beres et al. [35] develop a simulator specifically focused on fee and profitability[7], to empirically study LN's transaction fees and privacy provisions. Also Kappos et al. [39] develop a custom simulator to investigate the privacy of LN. Simulations are used to investigate the rouging problem, as well. Engelmann et al. [40] do it by considering channels characterized with economic-technical constraints. Zhang et al. [41] evaluate their routing algorithm, which is aimed to minimize the transaction fee of a payment path, subject to the timeliness and feasibility constraints. The source code of all these simulators is not public. Conversely, Brânzei et al. [42] disclose their custom sequential simulator, which is used to investigate the Bitcoin ecosystem economics taking into account off-ledger transactions and miner fees as incentives for honesty. Rebello et al. [36] introduce an open-source PCN simulator that implements the official LN message protocol in OMNET++. Recently, Conoscenti et al. [12] proposed CLoTH, an open-source simulator that reproduces the code of LND, with specific regards to routing and HTLC mechanisms. Different research works use CLoTH for evaluating their contributions (e.g., [13, 14, 15]).

6. Conclusion

In this paper, we investigated how a blockchain impacts on a second layer PCN built on top of it. To this end, we extended our PDES for SH-PCNs to incorporate the dynamics of the layer-1 blockchain having limited block size and not negligible block time. Then, we conducted an extensive evaluation and showed how blockchain parameters—such as its congestion rate—significantly influence PCN performance (e.g., in terms of payment success rate).

As future work, we will further develop our simulator by implementing key PCN features (e.g., routing, channel rebalancing techniques, network load). We also plan to conduct experiment on very large scale networks, up to a scale of 1:1 with the population of the euro area.

References

[1] M. Benedetti, F. D. Sclavis, M. Favorito, G. Galano, S. Giammusso, A. Muci, M. Nardelli, Self-balancing semi-hierarchical payment channel networks for central bank digital currencies, in: Proc. of 2024 IEEE PerCom Workshops, 2024, pp. 530–536. doi:10.1109/PerComWorkshops59983.2024.10503409.

[2] European Central Bank, What are instant payments?, https://bit.ly/3RolNlO, 2023. Accessed: 2023-11-10.

[3] M. Benedetti, F. De Sclavis, M. Favorito, G. Galano, S. Giammusso, A. Muci, M. Nardelli, PoW-less Bitcoin with Confidential Byzantine PoA, in: Proc. of 2023 IEEE Int. Conf. Blockchain and Cryptocurrency (ICBC), 2023, pp. 1–3. doi:10.1109/ICBC56567.2023.10174972.

[4] M. Benedetti, F. D. Sclavis, M. Favorito, G. Galano, S. Giammusso, A. Muci, M. Nardelli, Certified Byzantine Consensus with Confidential Quorum for a Bitcoin-derived Permis-

[6] https://github.com/rustyrussell/million-channels-project
[7] https://github.com/ferencberes/LNTrafficSimulator

sioned DLT, in: Proc. of the 5th Distributed Ledger Technology Workshop, 2023, pp. 1–17. Available at https://ceur-ws.org/Vol-3460/papers/DLT_2023_paper_1.pdf.

[5] A. Gangwal, H. R. Gangavalli, A. Thirupathi, A survey of layer-two blockchain protocols, J. Netw. Comput. Appl. 209 (2023) 103539. doi:10.1016/j.jnca.2022.103539.

[6] F. S. Mishkin, The economics of money, banking, and financial markets, Pearson education, 2007.

[7] P. Li, T. Miyazaki, W. Zhou, Secure balance planning of off-blockchain payment channel networks, in: Proc of IEEE INFOCOM '20, 2020, pp. 1728–1737. doi:10.1109/INFOCOM41043.2020.9155375.

[8] V. Sivaraman, S. B. Venkatakrishnan, M. Alizadeh, G. Fanti, P. Viswanath, Routing cryptocurrency with the spider network, in: Proc. of HotNets '18, ACM, 2018, p. 29–35. doi:10.1145/3286062.3286067.

[9] S. Z. Shabgahi, S. M. Hosseini, S. P. Shariatpanahi, B. Bahrak, Modeling Effective Lifespan of Payment Channels, 2022. ArXiv:2301.01240.

[10] N. Papadis, L. Tassiulas, Deep reinforcement learning-based rebalancing policies for profit maximization of relay nodes in payment channel networks, 2023. ArXiv:2210.07302.

[11] G. Galano, S. Giammusso, M. Nardelli, Modeling central bank digital currency over payment channels: A parallel ross- based approach, in: Proc. of 38th ACM SIGSIM Conference on Principles of Advanced Discrete Simulation (SIGSIM PADS '24), ACM, 2024. doi:10.1145/3615979.3656052.

[12] M. Conoscenti, A. Vetrò, J. C. De Martin, CLoTH: A Lightning Network Simulator, SoftwareX 15 (2021) 100717. doi:10.1016/j.softx.2021.100717.

[13] K. Asgari, A. A. Mohammadian, M. Tefagh, Dyfen: Agent-based fee setting in payment channel networks, 2022. ArXiv:2210.08197.

[14] V. Davis, B. Harrison, Learning a scalable algorithm for improving betweenness in the lightning network, in: Proc. of the 2022 Int. Conf. on Blockchain Computing and Applications (BCCA), 2022, pp. 119–126. doi:10.1109/BCCA55292.2022.9922233.

[15] T. Dasaklis, V. Malamas, Lightning network's evolution: Unraveling its present state and the emergence of disruptive digital business models, 2023. Preprints.org:202305.0523.v1.

[16] C. D. Carothers, D. Bauer, S. Pearce, ROSS: A high-performance, low-memory, modular Time Warp system, Journal of Parallel and Distributed Computing 62 (2002) 1648–1669. doi:10.1016/S0743-7315(02)00004-7.

[17] J. Poon, T. Dryja, The bitcoin lightning network: Scalable off-chain instant payments, 2016.

[18] E. Rohrer, J. Malliaris, F. Tschorsch, Discharged payment channels: Quantifying the lightning network's resilience to topology-based attacks, in: Proc. of 2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), 2019, pp. 347–356. doi:10.1109/EuroSPW.2019.00045.

[19] I. A. Seres, L. Gulyás, D. A. Nagy, P. Burcsi, Topological analysis of Bitcoin's Lightning Network, in: Mathematical Research for Blockchain Economy, Springer, 2020, pp. 1–12. doi:10.1007/978-3-030-37110-4_1.

[20] G. Avarikioti, Y. Wang, R. Wattenhofer, Algorithmic channel design, in: Proc. of 29th Int. Symposium on Algorithms and Computation (ISAAC 2018), volume 123 of *Leibniz International Proceedings in Informatics (LIPIcs)*, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2018, pp. 16:1–16:12. doi:10.4230/LIPIcs.ISAAC.2018.

16.
[21] D. J. Watts, S. H. Strogatz, Collective dynamics of 'small-world' networks, Nature 393 (1998) 440–442. doi:10.1038/30918.
[22] ECB, A stocktake on the digital euro, https://bit.ly/412yK8a, 2023.
[23] ECB Surveys, Study on the payment attitudes of consumers in the euro area (SPACE), https://bit.ly/412qbdE, 2022.
[24] M. Benedetti, F. De Sclavis, M. Favorito, G. Galano, S. Giammusso, A. Muci, M. Nardelli, Self-Balancing Semi-Hierarchical PCNs for CBDCs, 2024. ArXiv:2401.11868.
[25] E. Androulaki, G. O. Karame, M. Roeschlin, T. Scherer, S. Capkun, Evaluating user privacy in Bitcoin, in: Financial Cryptography and Data Security, Springer, Springer Berlin Heidelberg, 2013, pp. 34–51. doi:10.1007/978-3-642-39884-1_4.
[26] I. Eyal, E. G. Sirer, Majority is not enough: Bitcoin mining is vulnerable, Commun. ACM 61 (2018) 95–102. doi:10.1145/3212998.
[27] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, S. Capkun, On the security and performance of proof of work blockchains, in: Proc. of 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS '16), ACM, New York, NY, USA, 2016, p. 3–16. doi:10.1145/2976749.2978341.
[28] A. Deshpande, P. Nasirifard, H.-A. Jacobsen, eVIBES: Configurable and interactive ethereum blockchain simulation framework, in: Proc. of 19th Int. Middleware Conference (Posters), Middleware '18, ACM, New York, NY, USA, 2018, p. 11–12. doi:10.1145/3284014.3284020.
[29] A. Miller, R. Jansen, Shadow-Bitcoin: Scalable simulation via direct execution of Multi-Threaded applications, in: Proc. of 8th Workshop on Cyber Security Experimentation and Test (CSET 15), USENIX Association, Washington, D.C., 2015.
[30] M. Alharby, A. van Moorsel, Blocksim: A simulation framework for blockchain systems, SIGMETRICS Perform. Eval. Rev. 46 (2019) 135–138. doi:10.1145/3308897.3308956.
[31] C. Faria, M. Correia, BlockSim: Blockchain simulator, in: 2019 IEEE Int. Conf. on Blockchain, 2019, pp. 439–446. doi:10.1109/Blockchain.2019.00067.
[32] D. K. Gouda, S. Jolly, K. Kapoor, Design and validation of blockeval, a blockchain simulator, in: 2021 Int. Conf. on COMmunication Systems & NETworkS (COMSNETS), 2021, pp. 281–289. doi:10.1109/COMSNETS51098.2021.9352838.
[33] R. Paulavičius, S. Grigaitis, E. Filatovas, A systematic review and empirical analysis of blockchain simulators, IEEE Access 9 (2021) 38010–38028. doi:10.1109/ACCESS.2021.3063324.
[34] A. Albshri, A. Alzubaidi, B. Awaji, E. Solaiman, Blockchain simulators: A systematic mapping study, in: 2022 IEEE Int. Conf. on Services Computing (SCC), 2022, pp. 284–294. doi:10.1109/SCC55611.2022.00049.
[35] F. Beres, I. A. Seres, A. A. Benczur, A cryptoeconomic traffic analysis of bitcoin's lightning network, 2019. ArXiv:1911.09432.
[36] G. A. F. Rebello, G. F. Camilo, M. Potop-Butucaru, M. E. M. Campista, et al., PCNsim: A flexible and modular simulator for payment channel networks, in: Proc. IEEE INFOCOM Workshops '22, 2022, pp. 1–2. doi:10.1109/INFOCOMWKSHPS54753.2022.9798003.
[37] R. Yu, G. Xue, V. T. Kilari, D. Yang, J. Tang, CoinExpress: A fast payment routing mechanism in blockchain-based payment channel networks, in: Proc. of Int. Conf. on Computer Com-

munication and Networks (ICCCN), 2018, pp. 1–9. doi:10.1109/ICCCN.2018.8487351.

[38] G. Di Stasi, S. Avallone, R. Canonico, G. Ventre, Routing payments on the Lightning Network, in: Proc. of IEEE iThings and GreenCom and CPSCom and SmartData '18, 2018, pp. 1161–1170. doi:10.1109/Cybermatics_2018.2018.00209.

[39] G. Kappos, H. Yousaf, A. Piotrowska, S. Kanjalkar, et al., An empirical analysis of privacy in the Lightning Network, in: Financial Cryptography and Data Security, Springer, Berlin, Heidelberg, 2021, pp. 167–186. doi:10.1007/978-3-662-64322-8_8.

[40] F. Engelmann, H. Kopp, F. Kargl, F. Glaser, C. Weinhardt, Towards an economic analysis of routing in payment channel networks, in: Proc. of 1st Workshop on Scalable and Resilient Infrastructures for Distributed Ledgers (SERIAL), ACM, 2017, pp. 1–6. doi:10.1145/3152824.3152826.

[41] Y. Zhang, D. Yang, G. Xue, CheaPay: An optimal algorithm for fee minimization in blockchain-based payment channel networks, in: Proc. of the 2019 IEEE Int. Conf. on Communications (ICC), 2019, pp. 1–6. doi:10.1109/ICC.2019.8761804.

[42] S. Brânzei, E. Segal-Halevi, A. Zohar, How to charge lightning: The economics of Bitcoin transaction channels, 2022. ArXiv:1712.10222.